LA-UR- 03-1810

*Approved for public release; distribution is unlimited.*

*Title:*

# SIMPLE CLASSIFIERS FROM DATA DEPENDENT HYPOTHESIS CLASSES

*Author(s):*

Adam Cannon, 172300, CCS-3
James Howse, 119559, CCS-3
Donald D. Hush, 115295, CCS-3
James C. Scovel, 097403, CCS-3

*Submitted to:*

ICML conference
August 2003
Washington, D.C.

# Simple Classifiers from Data Dependent Hypothesis Classes

**Adam Cannon** CANNON@CS.COLUMBIA.EDU
Department of Computer Science, Columbia University, New York, NY 10027, USA

**James Howse** JHOWSE@LANL.GOV
**Don Hush** DHUSH@LANL.GOV
**Clint Scovel** JCS@LANL.GOV
Modeling, Algorithms, and Informatics Group, CCS-3, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

## Abstract

In this paper we introduce simple classifiers as an example of how to use the data dependent hypothesis class framework described by Cannon et al. (2002) to explore the performance/computation trade–off in the classifier design problem. We provide a specific example of a simple classifier and demonstrate that it has many remarkable properties. For example it possesses computationally efficient learning algorithms with favorable bounds on estimation error, admits kernel mappings, and is particularly well suited to boosting. We present experimental results on synthetic and real data that suggest that this classifier is competitive with powerful alternative methods.

## 1. Introduction

Consider the standard machine learning framework built around the Vapnik–Chervonenkis (VC) theory (Vapnik, 1998) and its treatment of the well known *error minimization* problem where we seek a classifier that minimizes the expected classification error. This framework provides performance guarantees for classifiers designed through empirical error minimization, but empirical error minimization is computationally hard for most nontrivial hypothesis classes. On the other hand restriction to trivial hypothesis classes alleviates the computational difficulties but does not provide good performance. These two extremes emphasize the importance of developing frameworks that facilitate the exploration of more moderate portions of the performance/computation space. Cannon et al. (2002) introduced a modification of the standard framework that enables this type of exploration in a nontrivial way. It involves an extension of the VC theory to the case where the hypothesis class depends on the data. In this new framework classifier design is decomposed into two components: the first is a restriction to a *data dependent* subclass of a hypothesis class and the second is empirical error minimization within that subclass. Exploration of the performance/computation trade–off is then performed in terms of the choice of *data dependent hypothesis class.* The study of how this choice affects performance is decomposed into two terms: *estimation error* which quantifies that portion of the error due to finite sample effects and *approximation error* which is the best error achievable by the data dependent class. To distinguish between hypothesis classes that are data dependent and those that are not we refer to the latter as "traditional classes". Cannon et al. (2002) introduced and analyzed several elementary examples of data dependent classes that were shown to fall between the two extremes. These examples motivate the introduction of *simple classes.*

We call a data dependent hypothesis class *simple* if the specific class realized by any given data set is the polynomial union of linearly ordered hypothesis classes, where a linearly ordered class is one whose indicator sets are linearly ordered by subset inclusion. We show how this definition facilitates the development of bounds on estimation error and computation. It also appears to facilitate the discovery of classes which are expressive enough to have good approximation error in practice. In Section 3 we define and analyze the *linear point–to–point* (LPP) data dependent hypothesis class as an example of a simple class. This function class has some remarkable properties. For example, in contrast to the intractability of empirical error minimization over traditional linear classifiers, LPP admits a low–order polynomial–time algorithm for empirical error minimization that is simple, numerically robust, fully parallelizable and has no free parameters. In addition we provide bounds on estimation error for LPP that are independent of dimension and similar in form to those obtained for a traditional class with VC dimension equal to 3. We also provide similar bounds on the difference between training and generalization error which is beneficial when it is important to have an accurate estimate of the actual performance of the classifier in the absence of a large test set. Because the estimation error bounds are independent of dimension, we are motivated to map to a higher dimensional space to improve performance, and since LPP classifiers are linear, computations can still be performed in the original space by employing kernel mappings. This situation is similar to support vector machines but here we optimize empirical error over a restricted class rather than margin over the unrestricted class. In summary, the simple class LPP when coupled with empirical error minimization successfully addresses nearly all the key issues in the classifier design problem.

The outstanding issue is approximation error, which we do not analyze theoretically but rather through simulations. Our results are consistent with the traditional framework in that they depend heavily on how well the class is matched to the process generating the data. We can address this issue in our framework by using elementary constructions to create additional simple classes. Cannon et al. (2003) construct alternative simple classes to demonstrate this observation. In this paper we only treat the LPP class but we consider the use of kernels and boosting (Schapire et al., 1998) as means for reducing approximation error. Kernelizing LPP is motivated in the above discussion about estimation error bounds. Boosting is especially suitable because LPP admits computationally efficient algorithms, even for *weighted* empirical error minimization. Boosting calls for the production of a base classifier at each round that minimizes a weighted empirical error. For most nontrivial hypothesis classes minimizing weighted empirical error is computationally intractable. Our empirical results demonstrate that even when LPP performs poorly on a particular problem instance it may be kernelized and/or boosted to a performance that is comparable to powerful methods such as support vector machines and random forests.

## 2. Error Minimization over Data Dependent Hypothesis Spaces

Consider a set $X$, the binary set $Y = \{0,1\}$ and a probability space $Z = X \times Y$. Let $z = (x, y)$ denote the corresponding random variable with probability measure $\mathcal{P}$, conditional probability measures $\mathcal{P}_y$ and $y$-marginal probability measure $P$. Let $\mathcal{F}$ denote a class of functions (classifiers) $f : X \to Y$ and let

$$e(f) = \mathcal{P}(f(x) \neq y)$$

denote the generalization error of the classifier $f$. Let $e^* = \inf_{f\in\mathcal{F}} e(f)$ denote the best error achievable in the class $\mathcal{F}$. We further suppose that we collect $n$ independent identically distributed (*i.i.d.*) samples $(z(1), z(2), .., z(n))$ from $\mathcal{P}$ and use them to construct an empirical error function

$$\hat{e}(f) = \frac{1}{n}\sum_{i=1}^{n} I\big(f(x(i)) \neq y(i)\big).$$

The work of Vapnik and Chervonenkis (1974) justifies the time honored learning strategy that chooses $\hat{f}$ to minimize $\hat{e}$ [1] by establishing the following probabilistic guarantee:

$$\mathcal{P}^n(e(\hat{f}) - e^* > \epsilon) \leq 8n^{V(\mathcal{F})}e^{-n\epsilon^2/128}, \qquad (1)$$

where $V(\mathcal{F})$ is the Vapnik–Chervonenkis dimension of the function class $\mathcal{F}$.

This result is only applicable when the the hypothesis class is chosen *before* data is observed. Cannon et al. (2002) showed that one could allow the hypothesis class to depend on the data and still obtain a bound on estimation error similar to (1). Following Cannon et al. (2002) we outline this result now. Let $z_n$ denote the $n$-sample consisting of individual samples $z_n(i), i = 1, .., n$. Given an $n$-sample $z_n$, we consider functions from a hypothesis space $\mathcal{F}_{z_n}$ which can depend on the $n$-sample and so is defined by a class $\mathcal{F}_n$ of functions on $(Z^n, Z)$. We define a data dependent class $\mathcal{F} = \{\mathcal{F}_n\}$ to be a collection of such classes $\mathcal{F}_n$. Next we define shatter coefficients for data-dependent function classes.

**Definition 2.1.** Let $\mathcal{N}_n(z_{2n}, \mathcal{F})$ be the number of distinct dichotomies of $2n$ points $z_{2n}$ generated by the function classes $\mathcal{F}_{z_n}$ where $z_n \subset z_{2n}$ varies over all size $n$ subsets of $z_{2n}$. That is, let $I_f = \{z : f(z) = 1\}$ denote the indicator set where the function is equal to

[1]In this paper we ignore questions of whether minima or maxima are actually attained. This detail is easy to include by introducing approximation parameters and approximate minima/maxima but obscures the presentation.

one. Then $\mathcal{N}_n(z_{2n}, \mathcal{F})$ is the number of different sets in

$$\{z_{2n} \cap I_f : f \in \mathcal{F}_{z_n}, z_n \subset z_{2n}\}.$$

We define the shatter coefficients as

$$S_{n/2n}(\mathcal{F}) = \sup_{z_{2n}} \mathcal{N}_n(z_{2n}, \mathcal{F}).$$

Theorem 2.1 (Cannon et al., 2002) below provides a bound on estimation error when using data dependent hypothesis classes. The proof of this theorem relied heavily on the main result from Cannon et al. (2002)

$$\mathcal{P}^n( \sup_{f \in \mathcal{F}_{z_n}} |e(f) - \hat{e}(f)| > \epsilon) \leq 2S_{n,2n}(\mathcal{F}_n)e^{2\epsilon}e^{-\frac{n\epsilon^2}{2}} \tag{2}$$

which bounds the error deviance.

**Theorem 2.1.** *Let $Y = \{0, 1\}$ and let $\mathcal{F}$ be a data-dependent classifier space. Given an i.i.d. $n$-sample $z_n$, let*

$$e^*_{z_n} = \inf_{f \in \mathcal{F}_{z_n}} e(f)$$

*denote the optimal generalization error in the data dependent class $\mathcal{F}_{z_n}$ and choose $\hat{f}$ to solve the learning strategy*

$$\min_{f \in \mathcal{F}_{z_n}} \hat{e}(f) \tag{3}$$

*of minimizing the empirical error over the class $\mathcal{F}_{z_n}$. Then for any $n$ and $\epsilon$,*

$$\mathcal{P}^n(e(\hat{f}) - e^*_{z_n} > \epsilon) \leq 2S_{n,2n}(\mathcal{F})e^{\epsilon}e^{-\frac{n\epsilon^2}{8}}.$$

In the next section we focus on a data dependent class we call linear point–to–point (LPP). The class LPP is a subset of the class of linear classifiers. When a data dependent class $\mathcal{F}$ is obtained by restricting a traditional class $\mathfrak{F}$

$$S_{n,2n}(\mathcal{F}) \leq (2n)^{VC(\mathfrak{F})} + 1.$$

Since the VC dimension for the class of linear classifiers in $\mathbb{R}^d$ is given by $d + 1$, we have

$$S_{n,2n}(\mathcal{F}_n) \leq (2n)^{d+1} + 1.$$

These bounds may be very loose. Indeed, Cannon et al. (2002) showed that the shatter coefficients for LPP are in fact independent of dimension and satisfy $S_{n,2n}(\mathcal{F}_n) \leq 8n^3 - 2n$. These bounds may be plugged directly into Theorem 2.1 to obtain probabilistic bounds on estimation error similar to (1).

It is interesting to note that for LPP the shatter coefficients satisfy (see Cannon et al. 2002, p. 348)

$$S_{n,2n}(\mathcal{F}_n) \leq (2n)^{VC_{2n}(\mathcal{F})} + 1,$$

where $VC_{2n}(\mathcal{F})$ is the data dependent $VC$ dimension of order $2n$ and is defined as

$$VC_{2n}(\mathcal{F}) = \max_{\{n \,:\, \exists\, z_n \subseteq z_{2n} \,:\, \mathcal{F}_{z_{2n}} \text{ shatters } z_n\}} n.$$

In words, $VC_{2n}(\mathcal{F})$ is the size of the largest subset of some $2n$ points which is shattered by the data dependent class on those $2n$ points. When the data dependent class $\mathcal{F}$ is obtained by restricting a traditional class $\mathfrak{F}$ then $VC_{2n}(\mathcal{F})$ will be less than or equal to the VC dimension $VC(\mathfrak{F})$.

## 3. A Simple Linear Classifier

We present a representative example of a simple classifier derived from a data dependent hypothesis class. The class we consider was introduced by Cannon et al. (2002) as a restriction of the traditional class of linear classifiers. For computational considerations we let $X = \mathbb{R}^d$ with the usual inner product and metric.

The *linear point–to–point* (LPP) data-dependent hypothesis class is the subset of linear classifiers whose orientations are determined by the pairwise differences between samples $\Delta^{ij}_n = x_n(i) - x_n(j), i \neq j$. For a fixed sample pair $(x_n(i), x_n(j))$, consider the family of linear classifiers defined by all hyperplanes orthogonal to $\Delta^{ij}_n$. The class of indicator sets on this family is linearly ordered by subset inclusion. The LPP class is the (polynomial) union of these families over all sample pairs in the training data. More formally LPP is the function class

$$\mathcal{F}_{z_n} = \{f : f(x) = \mathcal{H}(\Delta^{ij}_n \cdot x - b), i, j \leq n, b \in \mathbb{R}\} \tag{4}$$

for $i \neq j$ and where $\mathcal{H}$ is the heaviside function.

The ordering structure on the indicator sets is the defining characteristic of a simple class. Indeed exploiting this structure makes it an easy matter to design a polynomial–time learning algorithm that optimizes the empirical risk (3) over the class. Consider the following brute force approach where we are given an $n$-sample $z_n$ as input. For a fixed $\Delta^{ij}_n$ we compute and sort the dot products $v_k = \Delta^{ij}_n \cdot x_n(k)$ for $k = 1$ to $n$ and order the elements of the $n$-sample $z_n$ with respect to the sorted list. Now the dot products $\{v_k\}^n_{k=1}$ are used to determine thresholds $b_k = \frac{v_k + v_{k+1}}{2}$ for a sequence of $n - 1$ linear classifiers whose orientations are determined by $\Delta^{ij}_n$. Since the related indicator sets are ordered by subset inclusion and change incrementally at each new threshold, they are trivial to compute using the newly sorted $z_n$.[2] Similarly

[2]The extreme classifiers that label all points the same

we can use the ordering on the indicator sets to facilitate computation of the associated empirical errors and therefore to choose an empirically optimal $b^*$ with respect to $\Delta_n^{ij}$. So far computational expense is dominated by computing the dot products and sorting and is therefore $O(nd + n \log n)$. To find a globally optimal orientation and threshold pair we repeat this for all $O(n^2)$ differences $\Delta_n^{ij}$. The overall run time is then $O(n^3(d + \log n))$. Even a naive implementation of this brute force algorithm has many attractive properties. Firstly, it exactly minimizes the empirical error in a finite number of steps. Secondly, it uses only multiplications, additions and comparisons and is therefore likely to be more robust to finite precision computations than than other learning algorithms that employ operations such as division, linear solvers, and iterative solvers. Thirdly, it has no free parameters for the user to specify and analyze.

Since the LPP function class consists of linear classifiers, we can implement a *kernelized* version of the method in the usual way (e.g. see Cristianini and Shawe-Taylor 2000). Let $\phi : X \to \bar{X}$ be a map with a kernel $K(x_1, x_2)$ such that $\langle \phi(x_1), \phi(x_2) \rangle = K(x_1, x_2)$, where the dimension $\bar{d}$ of $\bar{X}$ may be much larger than $d$. It is straightforward to implement the LPP classifier in the mapped space $\bar{X}$ without having to compute in $\bar{X}$ by using the kernel to compute inner products. Employing a kernel map is attractive in that it results in a very minor change in the computational requirements, and does not affect the estimation error bounds for the class.

This method also has some very attractive properties when coupled with boosting. Recall that boosting is a procedure that, given a training set, produces an overall classifier that is a weighted majority vote of classifiers from a base hypothesis class. The procedure is unchanged by the utilization of a data dependent base hypothesis class provided the class is the same for each round of boosting. Moreover boosting strategies usually call for the determination of a base classifier at each round that minimizes a weighted empirical error. However, most classifier design algorithms used in practice *do not* minimize weighted empirical error because it is computationally intractable for the function classes they consider (*e.g.*, traditional linear classifiers), and so satisfaction of this objective is rarely guaranteed. On the other hand, with only minor modifications the brute force algorithm above is guaranteed to minimize weighted empirical error and therefore facilitates bonafide boosting procedures.

are not accounted for in this discussion but should be included when implementing this approach.

## 4. Experimental Results

The development of practical methods for classifier design involves a trade-off between computation, estimation error and approximation error. In the previous section we presented a computationally efficient learning strategy. The shatter coefficient bounds for this learning strategy give rise to favorable estimation error bounds that are independent of dimension. This fact is highlighted in what follows where we demonstrate that our approach is highly resistant to overfitting, even when the dimension is high and the sample size is small. The advantages of computational efficiency and favorable estimation error must be balanced against a possible sacrifice in approximation error. In principle, in the data dependent hypothesis framework, approximation error is primarily controlled through the choice of data dependency. Ideally this choice is based on first principles knowledge of the problem at hand.

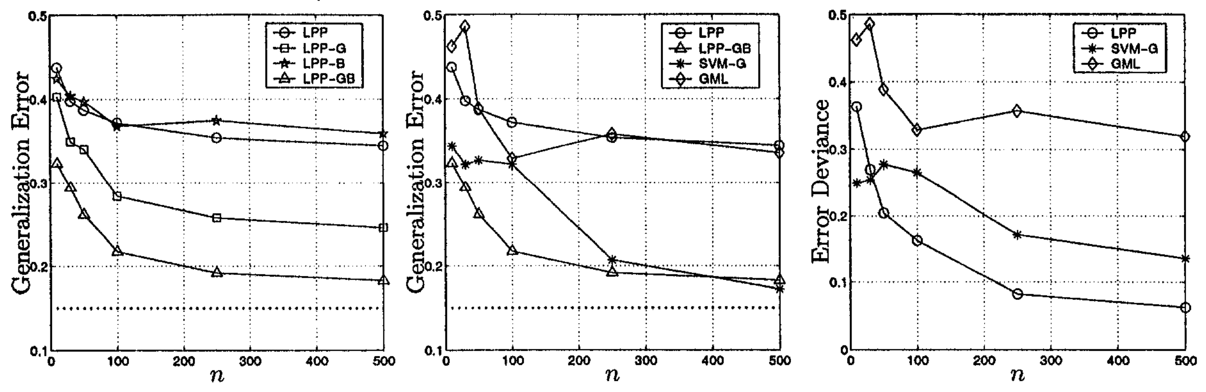
### 4.1. Synthetic Data

Our first set of experiments are performed on synthetic data generated from Gaussian distributions where we can compute the optimal classifiers and the values for the error minimization. We refer to the generalization error at the optimal measurable classifier as the Bayes' error $e_B$. In our experiments we report estimates of the average generalization error $E[e(\hat{f})] = \int e(\hat{f}) d\mathcal{P}$ of the learning strategy, where the average is over all training sets of a given size $n$ generated according to an *i.i.d.* sample plan.

We compare the LPP learning strategy with the Gaussian Maximum Likelihood (GML) learning strategy (e.g. see Fukunaga 1990). We choose GML because it has similar computational requirements and is a time honored method whose generalization properties are well known, particularly for the problem instances we have chosen for our experiments. Indeed, we expect the GML learning strategy to perform well in these experiments because it exploits the knowledge that the conditional class distributions are Gaussian. In this case the optimal classifier can be expressed as a quadratic classifier of the form

$$\mathcal{H}\Big((x - m_1) \cdot \Sigma_1^{-1}(x - m_1) - (x - m_0) \cdot \Sigma_0^{-1}(x - m_0) + \ln(|\Sigma_1|/|\Sigma_0|) + \tau\Big) \tag{5}$$

where the $m_i$ and $\Sigma_i$ are the conditional class means and covariances and the threshold $\tau$ is chosen so that $\tau = 2\ln(P(y = 1)/P(y = 0))$. The GML learning strategy produces a quadratic classifier by computing maximum likelihood estimates of $m_i$, $\Sigma_i$, substi-



(a) LPP: kernelized and boosted (b) LPP vs. other methods (c) Error deviance comparison

**Figure 1:** This figure summarizes the experimental results on the $I$–$\gamma I$ distribution.

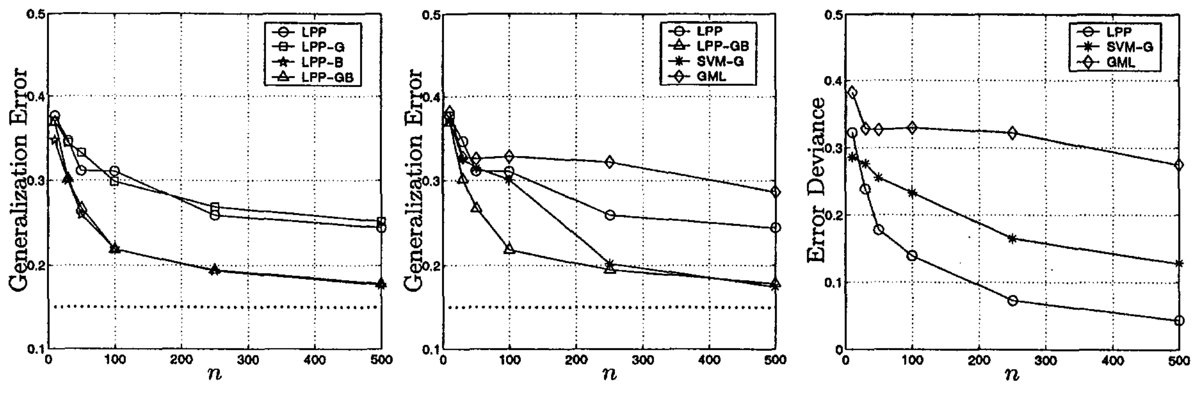
tuting them into (5), and then using maximum likelihood estimates of $\mathcal{P}(y = i)$ to determine $\tau$. To make this strategy complete we must specify its course of action when the estimated covariance matrices have reduced rank. In addition we would like to be robust to cases where the estimated covariance matrices are ill–conditioned. We address these concerns by inverting regularized covariance estimates of the form $\hat{\Sigma}_i + \beta I$ where $\hat{\Sigma}_i$ is the maximum likelihood estimate and $\beta > 0$ is the *regularization parameter* for our GML strategy. The computational requirements for this strategy are $O(nd^2 + d^3 + n \log n)$.

We also compare our simple classifier learning strategy with support vector machines. We choose support vector machines because, like the learning strategy for LPP, they possess estimation error bounds that are independent of dimension (Cristianini & Shawe-Taylor, 2000) and can be solved by a computationally efficient learning algorithm (e.g. see Hush and Scovel 2003). In particular we employ the soft margin support vector machine (SVM–SM) as described by Cortes and Vapnik (1995) with the specific $SVM^{light}$ learning algorithm (Joachims, 1999). This learning strategy has three parameters: The kernel function $K$, the positive real value $C$ that weights the slack variables in the SVM–SM criterion, and a positive real value *tol* associated with the stopping condition for the algorithm. The run times of SVM–SM algorithms tend to be more variable than the run times of simple classifier algorithms or GML algorithms. Although the run time guarantees for practical SVM-SM algorithms are as high as $O(C^2 n^5 \log n)$ (Hush & Scovel, 2003), empirical evidence suggests that the run time dependence on $n$ tends to be no higher than cubic (Joachims, 1999).

In our synthetic data experiments we set $d = 50$ and generate data according to probability distributions on $\mathbb{R}^d \times \{0, 1\}$ where the class conditional distributions $\mathcal{P}_0$ and $\mathcal{P}_1$ are Gaussian. We perform experiments with two different distributions whose parameters are chosen so that the Bayes' error is 0.15. For both distributions the class marginal probabilities are $P(y = 0) = 1/3, P(y = 1) = 2/3$. In the first distribution the class means are equal $\mu_0 = \mu_1 = \vec{0}$ and the class covariances are different $\Sigma_0 = 1.467I$ and $\Sigma_1 = I$. In the second distribution the class covariances are equal $\Sigma_0 = \Sigma_1 = I$ and the class means are different $\mu_0 = (0.276)\vec{1}$ and $\mu_1 = \vec{0}$. Following Fukunaga (1990) we refer to the first distribution as the $I$–$\gamma I$ distribution and the second as the $I$–$I$ distribution. Fukunaga (1990) suggests that practical learning strategies should be general enough to perform well on both of these distributions because in many classification problems the characteristic that distinguishes the two classes is some combination of mean difference and covariance difference.

For each learning strategy (e.g. LPP, SVM-SM, GML, boosting, etc.), and each training sample size $n = 10, 30, 50, 100, 250, 500$ the following is repeated $k = 20$ times. First, $n$ samples are randomly generated and used to train a classifier $\hat{f}$. Next, the error $e(\hat{f})$, of the resulting classifier is estimated using a test set of 50,000 random samples [3]. Finally, the estimated error is averaged over the $k = 20$ runs to obtain an estimate of $E[e(\hat{f})]$. Although the values we report are "estimates of averages", for expository purposes we drop the terms *estimate* and *average* and simply refer to

[3]The same test set of 50,000 *i.i.d.* random samples is used for all experiments.



(a) LPP: kernelized and boosted (b) LPP vs. other methods (c) Error deviance comparison

**Figure 2:** This figure summarizes the experimental results on the *I*–*I* distribution.

them as the *errors* or *generalization errors.*

The results for the $I$–$\gamma I$ distribution are illustrated in Figures 1(a)–1(c). In Figures 1(a)–1(b) the Bayes' error of $e_B = 0.15$ is shown as a dark dotted line. Figure 1(a) illustrates how the performance of a simple class that is not particularly well suited to this data distribution can be improved by adding a kernel and boosting. This figure plots the generalization error as a function of $n$ for LPP, LPP with Gaussian kernel (LPP-G), LPP with 500 rounds of boosting (LPP-B), and boosted LPP with Gaussian kernel (LPP-GB). The Gaussian kernel is $K(x_1, x_2) = e^{-\|x_1 - x_2\|^2/d}$ and LPP-GB uses the AdaBoost method described by Freund and Shapire (1997). These results demonstrate how the approximation error can be dramatically reduced by incorporating kernels and boosting.

In Figure 1(b) we compare the LPP classifier to some powerful alternative learning strategies. This figure plots the generalization error as a function of $n$ for LPP, LPP-GB, GML, and SVM-SM. The regularization parameter for GML was $\beta = 10^{-6}$. For the support vector machine we used a Gaussian kernel (same as above) and the default values of $C$ and *tol* in $SVM^{light}$ (i.e. $C = n / (\sum_{i=1}^{n} K(x_n(i), x_n(i)))$ and $tol = 0.001$). The results here are somewhat surprising. Firstly, the SVM-SM, LPP, and LPP-GB strategies all perform very well compared to GML even though they are not explicitly designed to exploit the knowledge that the probability distributions are Gaussian. Secondly, the LPP-GB strategy attains much lower generalization errors than SVM-SM for the smaller sample sizes. This demonstrates that by kernels and boosting a simple classifier strategy is able to attain generalization errors that are superior to some very powerful alternative methods. It is important to point out that these comparisons would be much different for larger sample sizes. Indeed, the performance of SVM-SM is already superior at $n = 500$ and for sufficiently large $n$ the performance of GML will also improve. Nevertheless it is striking that a simple classifier may be so clearly dominant at these smaller sample sizes.

These results are obtained with a simple classifier strategy that is more *stable* than GML and SVM-SM. We use the term *stable* to refer to the sensitivity of the performance of the learning strategy to changes in the training data. Formal definitions of stability are described by Bousquet and Elisseeff (2002) where they are used to study the performance of support vector machines. In particular they show how to obtain bounds on the average classifier error deviance $E[|e(\hat{f}) - \hat{e}(\hat{f})|]$ as a function of stability so that learning strategies with good stability have small error deviance. In Figure 1(c) we plot (an estimate of) the average classifier error deviance $E[|e(\hat{f}) - \hat{e}(\hat{f})|]$ as a function of $n$ for LPP, GML and SVM-SM. The error deviance is substantially smaller for the LPP learning strategy. Such results are predicted by the error deviance bounds in Equation 2. This characteristic of the simple classifier learning strategy means that we have higher confidence in our estimates of generalization error. More precisely it means that the true generalization error is known to fall within a smaller range with higher confidence. This is particularly useful in applications where a large test set is not available and it is important to have an accurate estimate of performance.

Results for the same set of experiments with the *I*–*I*

| | Ionosphere | Diabetes | Tuft's Nose |
|---|---|---|---|
| LPP | 19.0/11.4/***6.50*** | ***23.7***/24.0/26.7 | 16.6/16.5/***9.8*** |
| SVM-SM | 13.0/***9.1*** | 22.8/***22.7*** | ***16.8***/16.9 |
| GML | 12.5 | 23.9 | - |
| RF | 6.60 | 23.2 | 10.1 |

**Table 1:** Estimates of the average generalization value for LPP, GML, SVM-SM and random forests (RF). LPP results have three entries A/B/C where A is the value for the basic class, B is the value for the class with quadratic kernel and C is the value for the class with quadratic kernel and 500 rounds of boosting (except for the Tuft's nose data where 250 rounds of boosting is used). The reported results are for the final round of boosting. The SVM-SM has two entries A/B where A is the linear support vector machine and B is the support vector machine with quadratic kernel. No result is provided for GML on the Tuft's nose data because the dimension of the data is too high to obtain meaningful results with this method.

distribution are illustrated in Figures 2(a)–2(c). The parameters for the GML and SVM-SM learning strategies are the same as our first experiment. These results again demonstrate that the simple classifier strategy is competitive with powerful alternative methods. In Figure 2(c) we plot the average classifier error deviance $E[|e(\hat{f}) - \hat{e}(\hat{f})|]$ as a function of $n$ for LPP, GML and SVM-SM. Again we see that the error deviance is substantially smaller for the simple class LPP.

### 4.2. Real–World Data

This section describes experiments with three different real–world data sets. The first two are the *ionosphere* and *Pima Indian diabetes* data sets from the UCI repository (Blake & Merz, 1998) and the third is an *artificial nose* data set collected at Tufts University (Dickinson et al., 1996; White et al., 1996). The ionosphere data set consists of $n = 351$ samples of radar signals, $n_0 = 126$ with label $y = 0$ that passed through the ionosphere and $n_1 = 225$ with label $y = 1$ that did not pass through the ionosphere. Each radar signal contains $d = 34$ real valued measurements. The Pima Indian data set consists of $n = 768$ samples each containing $d = 8$ measurements. While it is often asserted that this data set has no missing values, Ripley (1996) comments that this is not true. We follow Ripley's recommendations reducing the data set to $n = 532$ samples with $d = 7$ measurements each. This reduced data set contains $n_0 = 355$ samples with label $y = 0$ (no diabetes) and $n_1 = 177$ samples with $y = 1$ (diabetes). The third data set is taken from an artificial nose developed at Tufts University. The nose consists of 19 optical fibers each of which has been coated on one end with a different organic dye. The data consists of the change in emission fluorescence intensity over time for each fiber. The change in intensity is measured at both the 620nm and 680nm wavelengths in each fiber. A 20 second time interval is sampled at 60 equally spaced points for each wavelength in each fiber. Hence each record consists of 38 observations each of which contains 60 samples, so each data sample contains $d = 2280$ measurements. Data samples were collected by exposing the fiber bundle to a 4 second pulse of a particular compound or mixture of compounds. The data set contains $n_0 = 760$ samples where the mixture contained trichloroethylene (TCE) and $n_1 = 352$ samples where the mixture contained no TCE, for a total of $n = 1112$ samples.

For each learning strategy the following is repeated $k = 100$ times for the ionosphere and diabetes data and $k = 25$ times for the nose data. The data set is randomly partitioned into two subsets: a training set containing approximately 2/3 of the data and a test set containing the remaining 1/3. The learning strategy is applied to the training set and an estimate of the error of the resulting classifier is computed using the test set. The errors are averaged over the $k$ runs to obtain estimates of the average generalization error.

We compare learning strategies for all three data sets. For the ionosphere data Breiman (1999) reports a generalization error estimate of 5.5% for his random forest learning strategy. We implemented the same random forest learning strategy using Breiman's software and obtained an estimate of 6.6% with our error estimation procedure. On the reduced Pima Indian data set Ripley (1996) obtained a generalization error estimate of about 20% for the best classifier, and he performed a diagnostic which he claims lower bounds the Bayes error for this problem at about 15%. Using the nose data set Priebe (2001) estimates a generalization error of 12.5% for the best $k$-nearest-neighbor strategy, and 4.5% for a generalized Wilcoxon-Mann-Whitney classifier.

We report our results for the LPP learning strategy along with GML, SVM-SM and random forests (RF). For GML we used a regularization parameter $\beta = 10^{-6}$. For support vector machines we used the

$SVM^{light}$ learning algorithm described by Joachims (1999) with default settings for the parameter values. LPP classifiers and support vector machines were employed in both their basic form and with the quadratic kernel $K(x_1, x_2) = (\langle x_1, x_2\rangle+1)^2$. LPP with quadratic kernel was boosted 500 rounds using the AdaBoost method (Freund & Shapire, 1997). We used the random forest implementation Forest-RI (Breiman, 1999) with number of trees $t = 1000$ and number of variables per node $F = 4$ for the ionosphere and diabetes data sets, and $t = 500$ and $F = 20$ for the nose data. The results are shown in Table 1.

It is instructive to consider what can be achieved with the nose data by the restricted class LPP compared to traditional linear classifiers. It is no surprise that this data set of 1112 samples in 2280 dimensions is linearly separable. Nearly all training algorithms for traditional linear classifiers are designed to produce a separating solution when one exists. Of all the linear classifiers that separate the data the SVM-SM solution is generally considered among the best. On the other hand the LPP solution, which does not separate the data, provides almost identical performance. The LPP strategy has the additional benefit that it has a lower error deviance which means that estimates of its performance are more accurate. The nonseparability of the LPP solution also means that boosting can be employed to improve performance, which is not true for traditional linear classifier learning strategies that produce separating solutions [4]. This is significant in the nose data because boosting LPP leads to a performance that is significantly better than SVM-SM.

## References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research, 2*, 499–526.

Breiman, L. (1999). *Random forests* (Technical Report 567). University of California - Berkeley, Berkeley, CA.

Cannon, A., Ettinger, M., Hush, D., & Scovel, C. (2002). Machine learning with data dependent hypothesis classes. *Journal of Machine Learning Research, 2*, 335–358.

Cannon, A., Howse, J., Hush, D., & Scovel, C. (2003). *Simple classifiers* (Technical Report LA-UR-03-0193). Los Alamos National Laboratory.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273–297.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods.* Cambridge ; United Kingdom: Cambridge University Press. 1st edition.

Dickinson, T., White, J., Kauer, J., & Walt, D. (1996). A chemical-detecting system based on a cross-reactive optical sensor array. *Nature, 382*, 697–700.

Freund, Y., & Shapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*, 119–139.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition.* San Diego, CA: Academic Press. 2nd edition.

Hush, D., & Scovel, C. (2003). Polynomial-time decomposition algorithms for support vector machines. *Machine Learning, 51*, 51–71.

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schlkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods - support vector learning.* MIT Press.

Priebe, C. (2001). Olfactory classification via interpoint distance analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*, 404–413.

Ripley, B. (1996). *Pattern recognition and neural networks.* Cambridge, UK: Cambridge University Press.

Schapire, R., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics, 26*, 1651–1686.

Vapnik, V. N. (1998). *Statistical learning theory.* New York NY: John Wiley & Sons, Inc.

Vapnik, V. N., & Chervonenkis, A. (1974). *Theory of pattern recognition.* Moscow: Nauka. (in Russian).

White, J., Kauer, J., Dickinson, T., & Walt, D. (1996). Rapid analyte recognition in a device based on optical sensors and the olfactory system. *Analytical Chemistry, 68*, 2191–2202.

[4]Boosting effectively terminates on the first round when the base classifier achieves zero training error.